ShadowMove: A Stealthy Lateral Movement Strategy

Amirreza Niakanlahiji, University of Illinois Springfield
Jinpeng Wei, UNC Charlotte
Md Rabbi Alam, UNC Charlotte
Qingyang Wang, Louisiana State University
Bei-Tseng Chu, UNC Charlotte
Lateral Movement Techniques

• **Advanced Persistent Threat (APT)** attackers use various lateral movement techniques

• Real world example: Equifax breach

• Features of **lateral movement** during APT attacks
  • Find a foothold within target networks
  • Use the compromised systems as stepping stones to reach critical systems

* Based on ZDNet article (https://zd.net/32Aqfol)
# Existing Lateral Movement Techniques

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploit vulnerabilities in network services</td>
<td>Increasingly hard due to advances in defense mechanisms</td>
</tr>
<tr>
<td>Harvest and abuse user credentials (e.g., passwords by Equifax breach)</td>
<td>Requires new network connections, which can be detected as anomaly</td>
</tr>
<tr>
<td>Inject application- and protocol-specific code into legitimate clients to reuse their connections</td>
<td>Complex and can be detected by existing defensive solutions (e.g., Windows Defender ATP)</td>
</tr>
</tbody>
</table>
Novelty of ShadowMove Attacks

• **No new connection, no extra authentication**: the attack process secretly reuses authenticated connections and injects commands through such connections.

• **No privilege elevation**: the attack is against client processes run by normal users.

• **No process injection (on Windows)**: the attack process secretly duplicates sockets owned by legitimate client processes without injecting code.

• **Application agnostic**: No prior knowledge about the target process is needed.
Overview

Client sends a request to the remote server
Client receives the response from the server
ShadowMove duplicates the socket created by the client
Overview

ShadowMove sniffs responses by peeking from the duplicated socket.
ShadowMove suspends the client, before sending requests to the remote server
ShadowMove sends a set of requests to perform an action.
Example of an action: Upload, Download, or Execute a file.
ShadowMove sends a set of requests to perform an action
Example of an action: Upload, Download, or Execute a file
Socket Duplicator

- Socket duplication requires cooperation of socket owner

1. WSASocket and WSAConnect
2. Get process id
3. Call WSADuplicateSocket
4. Send WSAPROTocol_INFO
5. Call WSASocket
6. Use duplicated socket
7. Close socket
## Socket Duplicator

- In ShadowMove, no co-operation is required

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>kernel/ntdll functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open the owner process with PROCESS_DUP_HANDLE</td>
<td>OpenProcess(PROCESS_DUP_HANDLE, , pid)</td>
</tr>
<tr>
<td>2</td>
<td>Foreach handle with type 0x24 (file)</td>
<td>NtQuerySystemInformation(SystemHandleInformation, ...)</td>
</tr>
<tr>
<td>3</td>
<td>Duplicate the handle</td>
<td>NtDuplicateObject</td>
</tr>
<tr>
<td>4</td>
<td>Retrieve its names</td>
<td>NtQueryObject(ObjectNameInformation)</td>
</tr>
<tr>
<td>5</td>
<td>Skip if the name is not \device\afsd</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Obtain remote IP and remote port number</td>
<td>getpeerrname(handle, ...)</td>
</tr>
<tr>
<td>7</td>
<td>Skip if remote IP and port do not match the input parameters</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Call WSADuplicateSocketW to get a special</td>
<td>WSADuplicateSocketW(handle, ...)</td>
</tr>
<tr>
<td></td>
<td>WSAPROTOCOL_INFO structure</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Create a duplicate socket</td>
<td>WSSocketW(WSAPROTOCOL_INFO, ...)</td>
</tr>
<tr>
<td>10</td>
<td>Use the socket</td>
<td>recv(), send()</td>
</tr>
</tbody>
</table>
Connection Detector

• Periodically gets a list of TCP connections
  • E.g. by calling GetTcpTable2 and GetTcp6Table2
• Identifies new connections
• Filters out the ones owned by a process that cannot be accessed
• Calls socket duplicator to duplicate the new ones
Peer Handler

- Receives network views from neighboring nodes
  - Peeks from duplicated sockets
  - waits for synchronization signal
- Sends synchronization signal periodically to its predecessor/successor nodes

Helps to construct a global view of the compromised network by synchronizing its current view with neighboring ShadowMove instances
Lateral Movement Planner

• Formulates the next lateral movement action plan
  • Current network view
  • History of action plans performed by all ShadowMove instances

An action plan describes the action that must be performed on a specific end point
Lateral Movement Planner

commitExecuteOperation(X, Y) :-
  connected(X, Y, Z), capability(Z, execute), origin(I),
  remoteOperation(I, Y, upload, _R),
  committed(_K, Y, upload).

remoteOperation( X, Y, Action, Route):-
  connected(X, Z, Service),
  capability(Service, Action),
  remoteOperation(Z, Y, Action, R),
  Route=[X| R].
Lateral Movement Plan Actuator

Contains a set of Protocol Handlers
  - Application protocol specific
    - FTP, TDS (MS SQL), and WinRM
  - Performs different operations
    - Upload, Download, or Execute

Creates application-specific queries to carry out lateral movement plans
Example Actuator Leveraging FTP

Compromised System

Remote System

ShadowMoveFTP

FTP Client

FTP Server

User testy

331 Please specify the password.

PASS 123456rtv @@

230 Login successful.

CWD /files/

250 Directory successfully changed.

TYPE I

200 Switching to binary mode.

PASV

Entering Passive Mode (54, 36, 162, 222, 176, 251)

STOR PoC2.txt

Entering Passive Mode
ShadowMove Implementation

• We implement prototypes of the ShadowMove design on Windows (2,501 lines in C/C++) and Linux (1,316 lines in C/C++)

• The lateral movement planner is based on SWI-Prolog

• A demo video of our ShadowMove prototype that leverages FTP is available\(^1\)

• The prototype implementation is available upon request (aniak2@uis.edu)

\(^1\)http://54.36.162.222/ShadowMoveDemo/ShadowmovePrototypeDemo.mp4
Why is ShadowMove Possible?

• The conflicting requirements between **process isolation and resource sharing** in commodity OS
  • allows the attack process to duplicate (share) sockets belonging to legitimate client processes.

• **A lack of built-in message origin integrity validation** in many networking protocols
  • allows malicious packets in existing connections that cannot be differentiated from legitimate packets.
Evaluation

• Not detected by off-the-shelf solutions

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Version</th>
<th>Update time</th>
<th>FTP/MSSql/WinRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AV</td>
<td>McAfee</td>
<td>16.0</td>
<td>03 Feb 2019</td>
<td>N/N/N</td>
</tr>
<tr>
<td>AV</td>
<td>Norton</td>
<td>22.16.2.22</td>
<td>03 Feb 2019</td>
<td>N/N/N</td>
</tr>
<tr>
<td>AV</td>
<td>Webroot</td>
<td>9.0.24.37</td>
<td>03 Feb 2019</td>
<td>N/N/N</td>
</tr>
<tr>
<td>AV</td>
<td>Bitdefender</td>
<td>6.6.7.106</td>
<td>03 Feb 2019</td>
<td>N/N/N</td>
</tr>
<tr>
<td>AV</td>
<td>Windows Defender</td>
<td>4.18.1901.7</td>
<td>03 Feb 2019</td>
<td>N/N/N</td>
</tr>
<tr>
<td>IDS</td>
<td>Snort (Windows and Linux)</td>
<td>2.9.12</td>
<td>07 Feb 2019</td>
<td>N/N/N</td>
</tr>
<tr>
<td>HIDS</td>
<td>OSSEC (Linux)</td>
<td>3.4.0</td>
<td>12 Oct 2019</td>
<td>N/--/--</td>
</tr>
<tr>
<td>HIDS</td>
<td>Osquery (Linux)</td>
<td>4.0.2</td>
<td>24 Oct 2019</td>
<td>N/--/--</td>
</tr>
<tr>
<td>HIDS</td>
<td>Wazuh (Linux)</td>
<td>3.10.2</td>
<td>24 Oct 2019</td>
<td>N/--/--</td>
</tr>
<tr>
<td>EDR</td>
<td>Cisco AMP</td>
<td>6.1.5.10729</td>
<td>14 Jun 2018</td>
<td>N/N/N</td>
</tr>
<tr>
<td>EDR</td>
<td>CrowdStrike Falcon Prevent</td>
<td>4.20.8305.0</td>
<td>11 Feb 2019</td>
<td>N/N/N</td>
</tr>
</tbody>
</table>
Limitations of ShadowMove Prototype

• It cannot hijack connections for which user-level encryption is applied to the payload

• It may not be able to get information such as the shellID in WinRM attack from the receiving buffer if the legitimate client consumes the buffer first

• Our design of ShadowMove on Linux relies on code injection
Thank you

Questions?

Amirreza Niakanlahiji, aniak2@uis.edu
Jinpeng Wei, jwei8@uncc.edu
Md Rabbi Alam, malam5@uncc.edu
Qingyang Wang, qywang@csc.lsu.edu
Bei-Tseng Chu, billchu@uncc.edu